Universität Basel 14.10.2014
Seminar: Open-Source-Softwareentwicklung Fabian Rauschenbach

Handout: Wie schreibe ich ein FLOSS Programm?

Die 17 Regeln aus „The Art of Unix Programming“[1]:

- Regel der Modularität: Programmiere einfache Bestandteile und saubere Schnittstellen.
- Regel der Klarheit: Code soll selbsterklärend sein. Die Lesbarkeit sollte nie verschlechtert werden.
- Regel des Aufbaus: Programm soll unabhängig sein und zu anderen Programmen verknüpfbar sein.
- Regel der Trennung: Die Schnittstellen von der Verarbeitungslogik trennen.
- Regel der Einfachheit: Programme mit dem Ziel der Einfachheit.
- Regel der Sparsamkeit: Grosse Programme so gut wie möglich vermeiden.
- Regel der Transparenz: Programmiere mit dem Ziel der Transparenz (es ist klar was das Programm macht) und der Feststellbarkeit (von aussen sichtbar, dass das Programm funktioniert).
- Regel der Robustheit: Unerwartete Konditionen funktionieren wie unter normalen Konditionen.
- Regel der Darstellung: Kluge Datenstrukturen verkraften eine „dumme“ Programmlogik.
- Regel der kleinsten Überraschungen: Schnittstellen sollen möglichst wenig Überraschungen bieten.
- Regel der Ruhe: Hat das Programm nichts zu sagen, dann soll es schweigen.
- Regel der Reparatur: Absturz möglichst laut und schnell. Diagnosen müssen feststellbar sein.
- Regel der Wirtschaftlichkeit: Arbeitszeit ist teurer, spare sie auf Kosten der Rechenzeit.
- Regel der Generierung: Handarbeit führt zu Fehlern, nutze Programme die Code schreiben können.
- Regel der Optimierung: Zuerst lauffähige Prototypen, dann die Optimierung.
- Regel der Vielfalt: Setze deinem Programm keine Grenzen.
- Regel der Erweiterbarkeit: Entwirf dein Programm für die Zukunft.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“[2]:

- Jede gute Software wird von einem Entwickler geschrieben, der ein persönliches Problem lösen will.
- Gute Programmierer wissen, was sie schreiben müssen. Brillante wissen, was sie neu schreiben müssen (und was sie wiederverwenden können).
- Plane eine Version wegzuwerfen; du wirst es sowieso tun.
- Wenn du die richtige Einstellung hast, werden dich interessante Probleme finden.
- Wenn du das Interesse an einem Programm verlierst, ist es deine Pflicht, dieses einem kompetenten Nachfolger zu übergeben.
- Wenn du deine Benutzer als Mitprogrammierer betrachtest, ist dies der einfachste Weg zu schneller Verbesserung und effizientem Debugging.
- Veröffentliche früh. Veröffentliche häufig. Und höre auf die Benutzer.
- Mit einer hinreichend großen Gruppe von Betatestern und Mitentwicklern wird fast jedes Problem schnell erkannt und die Lösung von jemandem gefunden.
- Schlaue Datenstrukturen und einfacher Code (im englischen Original: „dumb code“) funktionieren viel besser als andersherum.
- Wenn du deine Betatester wie deine wertvollste Ressource behandelst, werden sie dies auch werden.
- Fast so gut wie eigene gute Ideen zu haben, ist es, gute Ideen von den Benutzern zu erkennen. Manchmal ist letzteres besser.

1 Eric Steven Raymond (2003): The Art of Unix Programming (PDF: http://www.catb.org/esr/writings/taoup/)

2 Eric Steven Raymond (2001): The Cathedral and the Bazaar (PDF: http://www.catb.org/esr/writings/cathedral-bazaar/)

Universität Basel
Seminar: Open-Source-Softwareentwicklung

14.10.2014
Fabian Rauschenbach

Handout: Wie schreibe ich ein FLOSS Programm?

- Meist entstehen die brillanten Lösungen aus der Erkenntnis, dass das Problem falsch verstanden wurde.
- Perfektion (im Design) ist nicht erreicht, wenn man nichts mehr hinzufügen kann, sondern wenn nichts mehr entfernt werden kann.
- Jedes Tool soll so funktionieren, wie erwartet. Aber ein wirklich gutes Tool führt zu Verwendungszwecken, an die du niemals gedacht hättest.
- Wenn du Schnittstellencode schreibst, verhindere um jeden Preis, den Datenstrom zu verändern – und verwirf nur etwas, wenn dies der Empfänger verlangt.
- Wenn deine Programmiersprache überhaupt nicht Turing-vollständig ist, kann syntaktischer Zucker dein Freund sein.
- Ein Sicherheitssystem ist nur so sicher wie sein Geheimnis. Vermeide Pseudogeheimnisse.
- Um ein interessantes Problem zu lösen, suche eines.
- Mit genügend guter Kommunikation, wie über das Internet, und Führung ohne Zwang sind viele Köpfe immer besser als einer.

Truck Numbers:

- Statistischer Wert zwischen 1 und 0 (1 → sehr schelcht, 0 → sehr gut).
- Heroes, Spezialisten für ein spezifisches Code Segment. Sind besonders häufig in FLOSS Projekten.
- Contributors möglichst gut auf den Code verteilen
- Heroes einem komplett neuem/anderem Teil des Codes zuweisen